Parallel I/O on the IBM Blue Gene /L System

Authors:
ANL : Rob Ross
IBM : Jose Moreira
LLNL : Kim Cupps
SDSC : Wayne Pfeiffer

Abstract

Blue Gene/L is currently the world's highest performing and most scalable parallel system. With a raw compute power varying from 5.7 (single-rack configuration) to almost 370 Tflops (64-rack configuration), it can enable scientific and engineering computations that were just not possible before. A system with this much compute power can be very demanding on its I/O subsystem. Many important applications require the processing of vast volumes of data. With up to 1 TiB of main memory per rack, checkpoint/restart can also require the saving and restoring of a lot of data. This article focuses on the issue of supporting high-performance file I/O in the Blue Gene/L system.

1 Introduction

Blue Gene/L is an exciting new machine. With its unprecedented amounts of computing power and scalability, it enables scientific and engineering computations that have never been possible before. The Blue Gene/L system is different from most current parallel machines in that the compute nodes run a very light weight kernel called the compute node kernel (CNK). One feature of the CNK is that it forwards system calls off the compute node for service. A separate process, running on the IO node, services system calls on behalf of the CNK.

In this article we will discuss how parallel I/O is implemented on the Blue Gene/L machine and the available options for parallel file systems.

2 Background

A high-level block diagram of a Blue Gene/L system is shown in Figure 1. The organization of a Blue Gene/L system is based on the principles of physical partitioning and specialization. Compute nodes are specialized to run user application processes. The I/O nodes are specialized to perform system functions related to job control and I/O. The data servers are machines specialized to store data files used by the user applications.

The architecture of the data server cluster varies widely between Blue Gene system deployments, as we will see in this article. The data servers are typically implemented with conventional off-the-shelf servers. In some cases the data servers have the disks internal to them. In other cases, the data servers are connected to disks through a storage-area network (SAN). Depending on the file system operating in the Blue Gene/L I/O nodes, the data servers can be file servers (e.g., NFS servers for an NFS file system) or just storage servers (e.g., NSD servers for a GPFS file system).

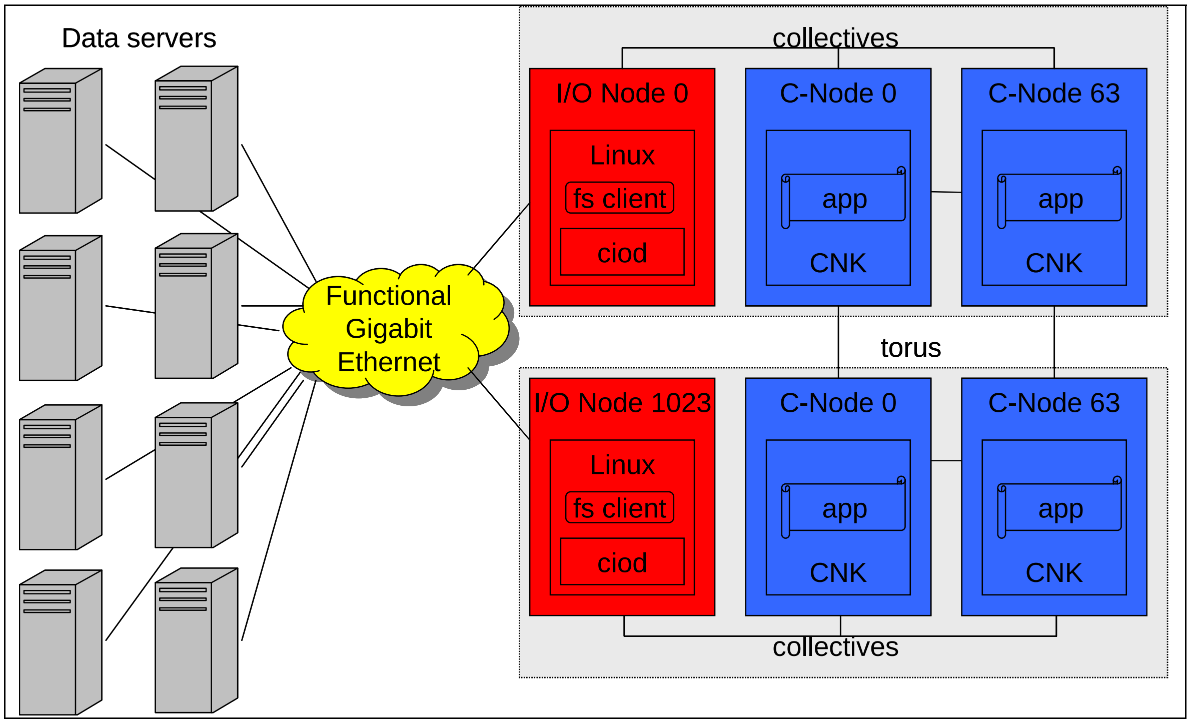

Figure 1: High-level block diagram of a Blue Gene/L system. It shows the compute and I/O nodes of the Blue Gene/L core and also the data servers that are connected to it.

User application processes run only on the compute nodes. Furthermore, the model is to have a single thread of execution per processor of the compute node. This can be accomplished either with a single process (in coprocessor mode) or with two independent processes (in virtual node mode) per compute node. (Note: In coprocessor mode, the second processor is used to perform message passing functions.)

The Compute Node Kernel (CNK) was designed to implement only the strictly necessary functionality for this model of execution, thus minimizing any perturbation of the running application processes. In particular, all I/O system calls initiated by user application processes running on a compute node (e.g., `open`, `close`, `read`, `write`) are trapped by the CNK and function shipped to its corresponding I/O node.

The I/O request shipped by CNK is received in the I/O node by a process called the *Control and I/O Daemon* (CIOD). This is a user process that runs with the same user id as the owner of the job running on the compute nodes. The CIOD simply reissues the request as a system call to the I/O node operating system. Because the I/O nodes are expected to perform more system functions (including I/O) we have chosen to run Linux as their operating system. This means that the I/O nodes can support any file system for which there is an appropriate port to Linux PowerPC 32-bit. BG/L machines may be configured with as few as one I/O node per 64 compute nodes, or as many as one I/O node per eight compute nodes.

Compute and I/O nodes are interconnected through the *collectives network*, also known as the tree network. This is the same network that supports reduction and broadcast operations among the compute nodes. In this case, it is used as a point-to-point network between a compute node and its corresponding I/O node.

I/O operations are always initiated by an application process. The CNK will compose a message describing the request and send it to the I/O node for execution by CIOD. Messages in the collectives network must be decomposed into individual packets of 256 bytes, and then reconstructed at the destination (CIOD for requests, CNK for replies). Most typical I/O operations can be implemented as single messages and many of them as single packets.

Very large file reads and writes, however, are decomposed into multiple messages, and each message is executed in order before proceeding to the next message. For example, a 5 MB read operation from the application process can be decomposed into 10 smaller read operations by the CNK. Each operation is then performed through a message that goes to CIOD for execution and then comes back with the resulting data. The CNK will complete all individual messages before finishing the read request and returning control to the application process.

The maximum size of messages between compute and I/O nodes can be controlled by the administrator for a particular installation through system configuration files. The CIOD environment variable `CIOD_RDWR_BUFFER_SIZE` controls the value of the buffer size for `read` and `write` system calls. The default value is 87600 bytes. “Popular” values that have shown good performance in installed systems, include 256 KiB and 512 KiB, although some groups use values as large as 2-4 MiB The various CIOD environment variables are summarized in Table 1.

Table 1: CIOD environment variables.

Variable name	**Description**
`CIOD_RDWR_BUFFER_SIZE`	The value is the buffer size for read and write system calls. The value can be increased or decreased to find the best match for the network and file servers used by the Blue Gene system. CIOD allocates one buffer of the specified size for each compute node that it is managing. A larger value causes CIOD to allocate more memory. The default value is 87600.
`CIOD_TREE_RECV_RETRY`	The value is the number of times CIOD tries to receive a packet from the tree device before deciding a packet is not available and yielding the processor. A value of 16 causes CIOD to never yield and return immediately. A larger value causes CIOD to pay more

	attention to the tree. The default value is 16.
`CIOD_TREE_SEND_RETRY`	The value is the number of times CIOD tries to send a packet to the tree device before yielding the processor. A value of 0 causes CIOD to never yield and loop until the packet is sent. A larger value causes CIOD to pay more attention to the tree. The default value is 0.
`CIOD_TREE_MULTIPLIER`	The value is the default multiplier controlling how often CIOD checks for messages from the compute nodes before checking the control connections and waiting I/O. The tree multiplier is dynamically adjusted by CIOD when at least one compute node is using a debugger or there is waiting I/O. A larger value causes CIOD to pay more attention to messages from compute nodes. The default value is 8192.
`DEBUG_SOCKET_STARTUP`	The value is a string that controls what blocks have the CIOD service connection enabled. The value ALL enables the service connection on every block. Otherwise, the value is the name of one block where the service connection is enabled. When the variable is not set, the service connection is not enabled.

CIOD supports a service connection that provides a trace facility and status reporting. The intent is to let a system programmer or administrator check on a running CIOD without disturbing it. CIOD starts a secondary thread that monitors the service connection so there is minimal impact to the work being done by the main thread. CIOD uses TCP port 7201 for the service connection. The trace facility supports turning tracing on and off for categories of events. By default, the trace output is sent back across the service connection but can also be directed to a file. The trace output allows you to see the flow of control through a running CIOD. The status reporting allows one to see the current state of various objects maintained by CIOD. You can check on individual compute nodes and get overview performance information for certain system calls.

As previously noted, the I/O nodes can use any file system that has been ported to their version of Linux for PowerPC 32-bit. As part of the boot process, every I/O node mounts from a primary NFS server that contains the code for the rest of the boot. Once that primary file system is mounted, the I/O nodes can mount other file systems. The rest of this article investigates the options for parallel file systems that are available for Blue Gene/L.

3 File Systems

There are currently four file system options being actively pursued for Blue Gene/L systems: NFS, GPFS, Lustre, and PVFS2.

3.1 NFS

The first option is to use NFS to mount a file system. This works with almost any file system, parallel or not. If a parallel file system is available, it ,may be NFS-exported from a variety of data storage machines, and each I/O node can NFS-mount from a different machine. This is a cheap and easy way to deploy a parallel file system for Blue Gene/L. However, there are scalability limitations associated with this configuration, since NFS is not really aware of the parallel nature of the underlying file system. In experiments with NFS-exported GPFS, we observed the performance to be quite good when each application process accesses a different file. On the other hand, the performance is very bad when multiple application processes share the same file.

3.2 GPFS

GPFS on Blue Gene – by Wayne Pfeiffer, San Diego Supercomputer Center

The second option is to use GPFS, which is the parallel file system developed and supported by IBM. The San Diego Supercomputer Center (SDSC) has been running a beta version of GPFS since April 2005. In December 2005, IBM released the first official version of GPFS for Blue Gene.

The most common configuration for a cluster running GPFS has the disks attached via so-called Network Shared Disk (NSD) servers, because this configuration allows operation over an IP networking infrastructure and has been shown to scale well. In a conventional cluster the compute nodes function as GPFS clients, whereas for Blue Gene the I/O nodes constitute the cluster and function as the clients. The compute nodes, which run only a lightweight kernel, use the tree network to move file I/O to and from the associated I/O nodes. SDSC's single-rack Blue Gene has 128 I/O nodes, the maximum allowable. This I/O-rich configuration enables high I/O performance.

Two specific parallel file systems are provided: bggpfs and gpfs-wan. Both are accessible via GPFS and use IA-64 nodes as the NSD servers. bggpfs is dedicated to Blue Gene. It has 12 NSD data servers, 2 NSD metadata servers, and 20 TB of disk. By contrast, gpfs-wan is shared with other supercomputers at SDSC as well as across the TeraGrid. gpfs-wan is much larger, with 58 data servers, 6 metadata servers, and 220 TB of disk. In each case the servers are attached via Gigabit Ethernet to the I/O nodes and via Fibre Channel to the disk. Between the I/O nodes and the NSD servers is a Gigabit Ethernet switch with a single link to each server. These links limit the peak performance possible with each file system.

Performance

I/O performance of both file systems has been measured with the IOR benchmark code from LLNL. Figure 2 shows the results of scaling scans obtained running in coprocessor mode with the default node mapping. Also shown are the peak rates possible with each file system. The peak rate curves are the same for both file systems up to 32 processors (32p). For 64p and higher, the peak rate using bggpfs is at a plateau of 1.5 GB/s set by the links to the 12 data servers. The peak rate curve using gpfs-wan increases beyond 32p until it reaches a plateau of 7.25 GB/s set by the links to the 58 data servers.

The steps in the peak rate curves arise when the ratio of compute nodes to I/O nodes increases for the default mapping. That ratio increases from 1 to 2 going from 1p to 2p, from 2 to 4 going from 8p to 16p, and from 4 to 8 going from 64p to 128p. The write rate curves for both file systems are between 50% and 65% of the corresponding peak rate curves over the entire processor range except for 1024p with gpfs-wan. The behavior of the read rate curves is more complicated

For gpfs-wan the write rate reaches a maximum of just over 4.0 GB/s on 512p and then drops somewhat to 3.2 GB/s on 1024p. By contrast, the read rate reaches a maximum of 2.2 GB/s on 1024p after a modest drop from 256p to 512p. For bggpfs the write and read rates are the same and constant at about 800 MB/s for 64p and above. For gpfs-wan at higher processor counts the write and read rates vary considerably from one IOR run. This is likely because this file system is shared with other systems. The rates shown are the best from several repeats. By comparison, the write and read rates for bggpfs vary relatively little from run to run.

The results discussed so far were obtained in coprocessor mode because that simplifies making scaling scans. Most users, however, run in virtual node mode, so additional results were obtained in that mode at the higher processor counts. For gpfs-wan the write and read rates on 512p and 1024p in virtual node mode are lower than those in coprocessor mode and reach final values of 3.1 GB/s for writes and 1.6 GB/s for reads on 2048p. For bggpfs the write and read rates on 512p and 1024p in virtual node mode are essentially the same as those in coprocessor mode and are sustained on 2048p.

In summary, rates of 4.0 GB/s for writes and 2.2 GB/s for reads have been achieved during shared use of gpfs-wan in coprocessor mode, while rates of about 75% of those have been achieved in virtual node mode.

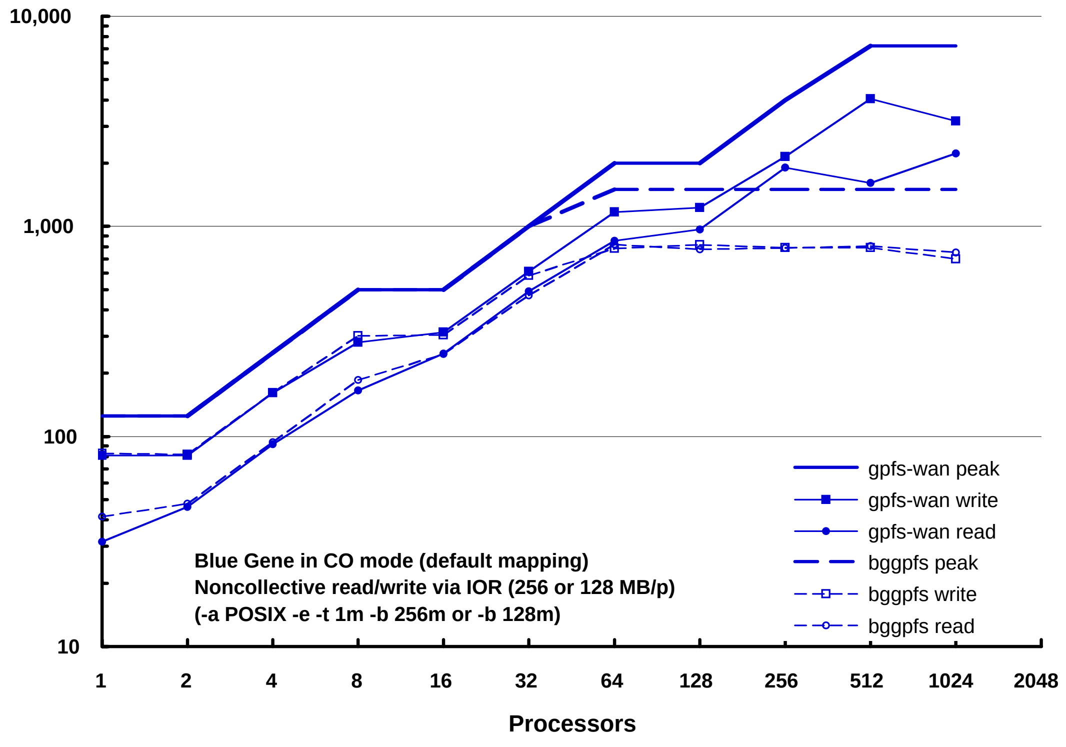

Figure 2: Benchmark results from BG/L system at SDSC running GPFS.

3.3 Lustre

The third option being pursued is to use CFS's Lustre file system. Lawrence Livermore National Laboratory's 64K Blue Gene/L system uses Lustre as its parallel filesystem. Lustre is a scalable cluster file system designed, developed and maintained by Cluster File Systems, Inc.. For more about CFS and Lustre in general please visit http://www.lustre.org/index.html. There is no public release of Lustre support for BG/L at this time.

LLNL Hardware Configuration

Livermore's 64 rack BGL system uses an I/O configuration of 64 compute nodes for every I/O Node (see Figure 3). BGL has 1K (1024) I/O nodes. Each I/O Node is a Lustre client and is Gigabit Ethernet attached to a system of 15 switches (the federated switch and BGL and Lustre edge switches, to be described later) that connects to the 896 Terabyte Blue Gene Lustre Cluster (BLC) file system via 448 Gigabit Ethernet links. BLC is a cluster of 224 Intel dual Nocona nodes with 2GB memory and 2.8GHz processors. Each of these Nocona nodes is a Lustre "Object Storage Server" (OSS). Each OSS has 2 Gigabit Ethernet links coming into it and connects to a Data Direct Networks 8500 Controller via a single 2Gb/s host interface. The 896 Terabytes of disk storage is provided by twenty eight racks of Data Direct Networks (DDN) RAID storage. Each rack of DDN storage contains 32 TB of usable SATA storage accessible via eight 2Gb/s fibre channel interfaces.

The link between BGL and storage is provided by a network of Cisco 6509 switches. Currently the 1024 1Gb/s ION interfaces associated with BGL and the storage purchased with it is attached to "edge switches". The edge switches are connected to a cluster of six "core" switches using 10Gb/s connections. Two other systems currently "share" the BLC file system by way of the federated network. The first of these is the Storage Lustre Interface Cluster (SLIC) a cluster of 10 Itanium Tiger 4 nodes that is used to transfer files out of the Lustre file system into the High Performance Storage System (HPSS) data archive for permanent storage. Users log into SLIC and either use parallel ftp or htar to move their files to HPSS. The other cluster with access to BLC is Gauss. Gauss is a high performance visualization cluster of 256 dual opteron (6 GB memory per CPU) Infiniband interconnected nodes, using NVIDIA 4500 graphics and each with a GE link into the federated switch that connects it to the Blue Gene Lustre Cluster file system. Gauss will be used exclusively by users of BGL for data visualization purposes.

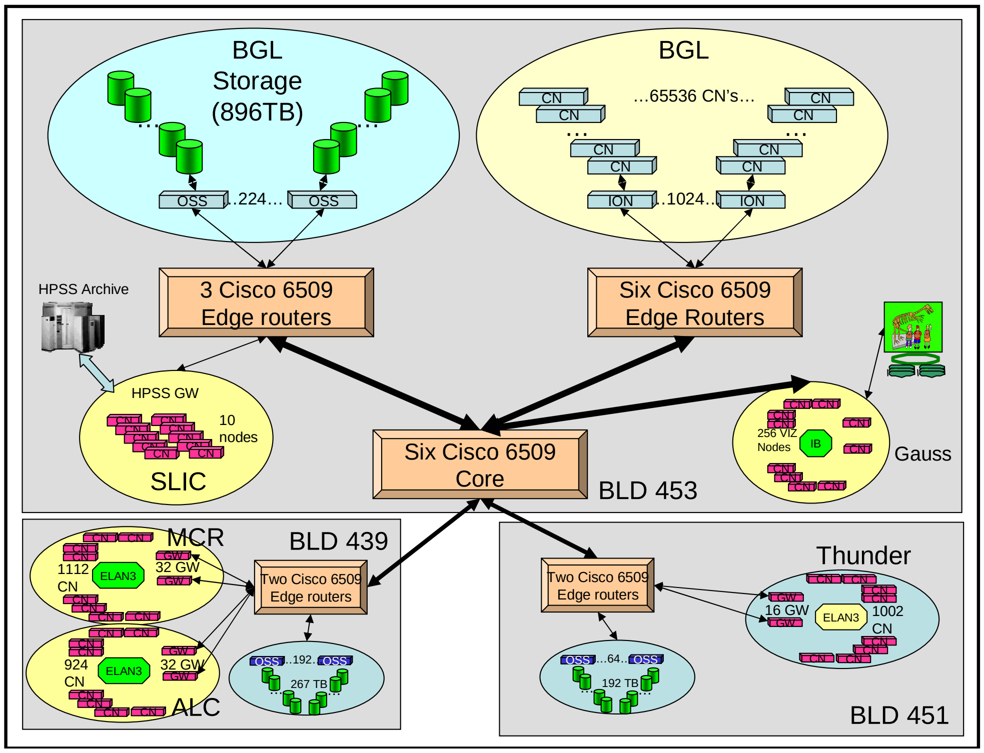

Figure 3: Lustre Storage Network Architecture at LLNL.

Projected Performance

With its 1024 1 Gb/s interfaces our BGL system has a potential I/O bandwidth of 128GB/s. Our original storage design specified an I/O bandwidth of 40GB/s which was far below our normal I/O bandwidth to CPU performance ratio. However after economic reality set in we were forced to scale back our goals. The adjustments began with the disk subsystems. We had a fixed storage budget and at the time the disks were purchased we had the choice of 146GB fibre drives for over $1000 each or enterprise class 250GB SATA drives for less than $500 each. Twice the storage for half the cost is a compelling argument. Our budget allowed us to purchase 28 racks of SATA storage. Each rack has a theoretical bandwidth of 1.2GB/s for a total of 33.6GB/s. However you don't get something for nothing and the trade-off we made was in terms of performance. SATA disks provide very good streaming performance but poor random performance. With a ratio of 65,536 (or 131,072 in virtual node mode) clients to 448 RAID devices our workload is random. Based upon the disk specifications we expect a request to take 18MS ~= 55 requests per second or given a 1MB block size about 55MB/s per RAID device. We would therefore expect to achieve around 24.8GB/s with our 448 RAID devices.

Our Lustre network was designed to provide 45GB/s theoretical bandwidth between BGL and its associated storage cluster. We use six Cisco 6509's as BGL "edge" switches (connecting the BGL core to the federated switch infrastructure) and three 6509's as disk edge switches (connecting the 224 file system OSS nodes to the federated switch). We tie the whole thing together with another six 6509's acting as core switches. The Lustre core architecture was also designed to provide a connection point for the other parts of our Lustre based "site wide global file system (SWGFS). The network architecture is based upon layer 3 switching, uses "equal cost multi-pathing" (ECMP) and link aggregation where appropriate. We hope the network can deliver 80% of theoretical bandwidth or 36GB/s.

Actual Performance

We tried to analyze the capabilities of each I/O related element in the path between the user application code and the physical disk devices independently. Then we tested groups of elements leading up to a user level I/O benchmark (IOR) designed to simulate actual user behavior. The lessons learned and problems found and fixed are far too extensive to discuss in this forum but the bottom line is that as expected, the disks have proven to be the limiting factor related to I/O performance. With exceptional cache management and carefully selected block sizes, write performance can exceed raw disk random I/O rates. The best full system write performance we have achieved is around 26GB/s. Improving read performance given the number of concurrent data streams per physical device, the amount of memory available in the I/O nodes for caching, and the synchronous nature of CIOD has proven to be a harder task. The best read performance we've achieved is around 22GB/s, but were not achieved in the run below. The results in Figure 4 were generated using the IOR benchmark on 11/25/05.

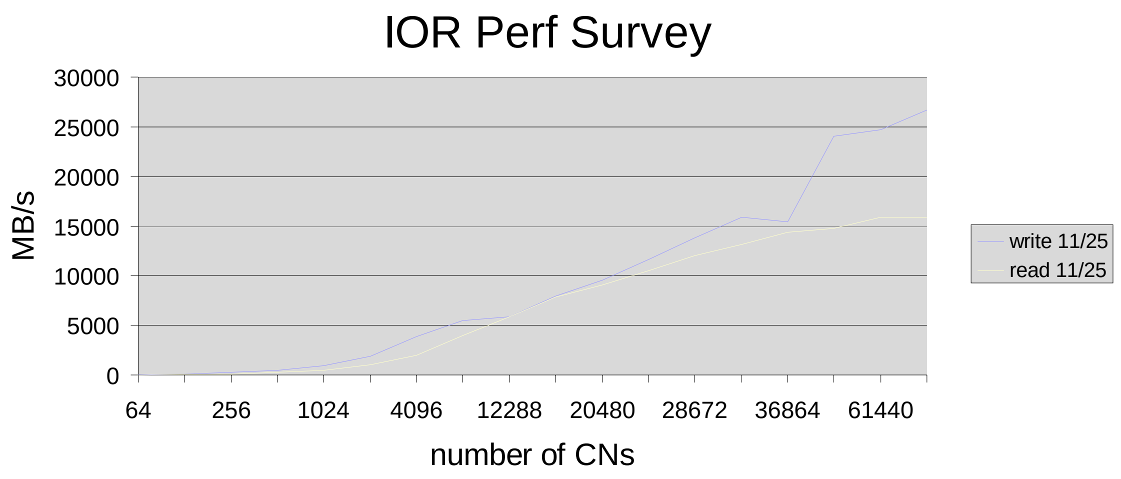

Figure 4: Benchmark results from BG/L system at LLNL running Lustre.

Summary
An enormous amount of work has gone into making Lustre perform on Livermore's BGL system. As stated previously, our approach was to isolate and tune each component and then combine and tune again. The conglomeration of components involved in I/O on BGL from the Lustre client and server software itself, to the tunables of the DDN controllers, and the network edge and core switches, as well as the buffering in the CIOD on the ION, have all been tweaked many times in the quest to improve performance. With each step forward we have taken at least half a step back, and that is on a good day. There is still work to do. All performance numbers listed in this study were run in co-processor mode. Performance in virtual node mode (the second processor is used for calculation tasks as well as communication) is substantially below co-processor mode performance. In fact, this is because we have spent almost no time studying or tuning virtual node mode performance. Metadata performance is also an area that requires further study and improvement as is simultaneous access to the Lustre file system from all systems. It has taken over a year to get Lustre performance and reliability to where it is today and we will continue to study and improve remaining performance and reliability issues.

3.4 PVFS2

The final option is PVFS2. PVFS2 is an open source parallel file system developed by researchers at Argonne, Clemson University, and Ohio State Supercomputer Center with help from a broad community of developers and users. While not officially supported by IBM, replacement IO node images are available to run PVFS2 on BG/L as part of the ZeptoOS project. PVFS2 is currently installed at ANL and at MIT, and it has been tested on the BGW system at IBM Watson.

The PVFS2 system uses a set of PVFS2 servers to allow PVFS2 clients to access the shared file system space. On BG/L, IO nodes are PVFS2 clients. The ZeptoOS IO node image provides the kernel module and user-space daemon that PVFS2 clients use to contact PVFS2 servers. BG/L compute nodes use the ciod system to forward I/O operations to IO nodes, just as with the other file system options. BG/L login nodes also mount the file system, so that users can access data stored on the PVFS2 file system between their application runs and move data on and off the file system conveniently.

The IO nodes use a Gigabit Ethernet network to communicate with PVFS2 servers. At ANL, 14 PVFS2 servers combine to provide a modest 6.7 Tbyte file system. Servers are xSeries 346 nodes, each with a 3.4GHz Xeon processor, 4 Gbytes of RAM, and 800 Gbytes of 10K SCSI HDDs on a ServeRAID 6i+ controller in RAID5 mode. The built-in GigE connections were replaced with Intel Pro 1000/MT GigE cards in order to allow for the use of jumbo frames. This configuration peaks at 1.2 Gbytes/sec read and 620 Mbytes/sec writes, and a single process can create approximately 50 files/second.

As part of the IBM BGW Consortium Day, the PVFS2 team had an opportunity to test PVFS2 on the BGW system. This system has both a larger number of compute nodes and was configured with more PVFS2 servers, 33, during testing. The disk resources on these servers were very modest, significantly limiting overall write performance. Results from this testing are shown in Figure 5.

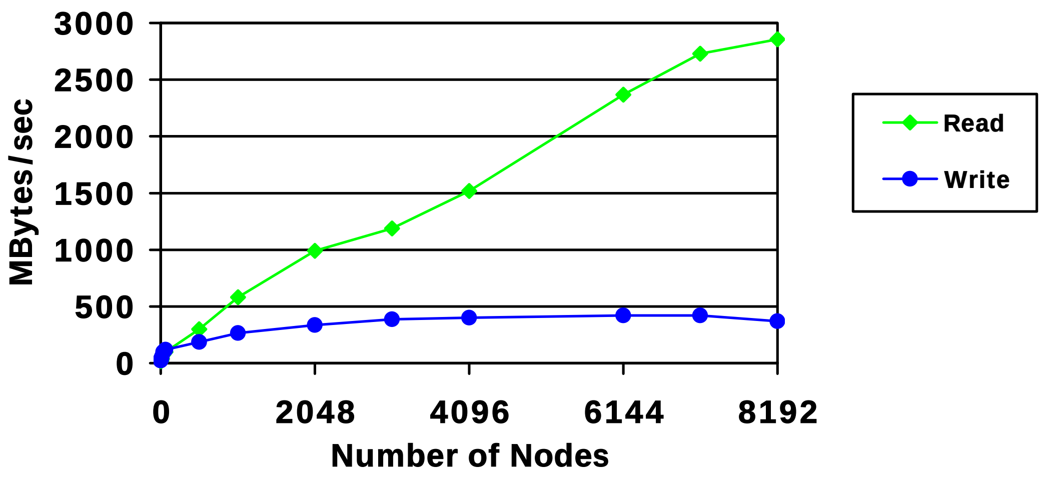

Figure 5: Benchmark results from BG/L system at IBM Watson running PVFS2.

PVFS2 is the first parallel file system to be made freely available for the BG/L system. Installation is not trivial, but an experienced system administrator should have no problems. For large I/O operations, performance is quite good, and the PVFS2 team is actively working to improve the performance of the system overall, including both small

I/O operations and metadata operations, such as listing directories and creating and removing many small files.

4 Conclusions

Multiple options are available for attaching storage to BG/L systems. For systems with very light I/O loads, an NFS solution is probably adequate. For heavier I/O loads, a parallel file system is preferable. If commercial support is a concern, IBM's GPFS is currently the best bet. For a free, community-supported solution, PVFS2 is a great option and can be matched with a wide variety of storage hardware. And in the near future the popular Lustre file system will be available as well.

5 References

ANL MCS	http://www-fp.mcs.anl.gov/division/welcome/default.asp
ANL BG system	http://bgservice.bgl.mcs.anl.gov/bgl/
LLNL	http://www.llnl.gov/asci/platforms/bluegenel/
SDSC	http://www.sdsc.edu/
IBM Research	http://www.research.ibm.com/bluegene/
Lustre	http://www.lustre.org/index.html
GPFS	http://www-03.ibm.com/servers/eserver/clusters/software/gpfs.html
PVFS2	http://www.pvfs.org/pvfs2/
ZeptoOS	http://www-unix.mcs.anl.gov/zeptoos/
BG Consortium	http://www-fp.mcs.anl.gov/bgconsortium/